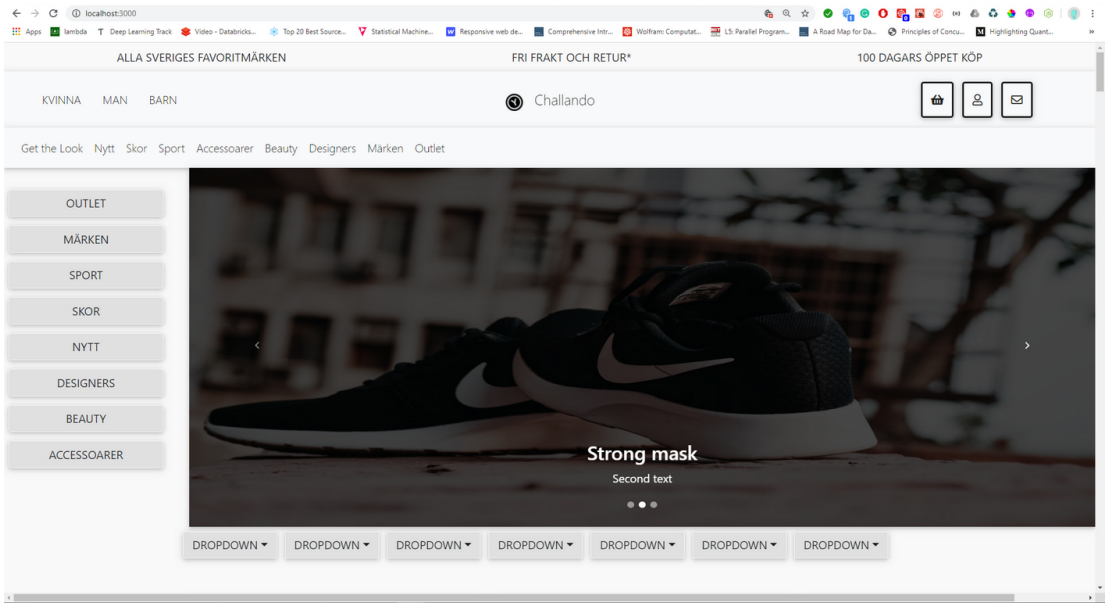


Figure 1: Main page

Dat076 Report- Grupp4
Challando E-handel

Madeleine Lexen Yazan Ghafir Philip Hellberg

March 2020

1 Introduction

The project is a fullstack web application for a clothing web shop. The applications backend is mainly implemented using the framework JakartaEE and has a JAVA Derby Database which is created with the Java Persistance API Hibernate. The frontend technology is React.js. The application is intended to work as two seperate microservices, with a separate backend and frontend. This was done in order to make the application more fault tolerant.

The backend is build, compiled and bundled via Maven and is deployed using

a Payara Application server. The frontend is compiled and run on Node npm server.

The purpose of the application is that the owner of the web shop should be able to add, change, remove and display their products to customers in the web application and for the customers to be able to browse, choose, pay and create an order.

1.1 Use Cases

1. Register a user to the website
2. Log in to the website
3. logout from the website
4. As a logged in user, visit My Page and see personal information
5. As a logged in user, browse previous orders in My pages
6. Browse the products
7. View product details
8. Select size for product
9. Add clothing item to cart
10. View cart
11. Change quantity of a clothing item in the cart
12. remove item from cart
13. proceed to checkout
14. and fill in billing address and payment information
15. complete purchase
16. Contact us forum, that saves the entered message, username, email and subject in the database to be reviewed by admins.
17. navigate to Login, Shopping cart, Contact us and Main page though the footer.

2 User Manual

User manual - Describe how to use your app, from the user's point of view

The user navigates to the front page, here the user can browse all the clothes in the web shop. If the user wants to see more details for a particular clothing item, the user can click in the image of the item, or the view button. The application the takes the user to a new page, where the details for the clothing

item are displayed. Here the user can select a size for the clothing item, and add it to cart.

In the navigation bar there are symbols for shopping cart, profile page and contact form, which the user an navigate to by clicking on the symbols. In the cart, the user can see the items that they've added, how many they've added and what the total cost will be for the cart if they were to order it. The user can change the quantity for each item in the cart, as well as remove it from the cart. There's also a button for proceeding to checkout.

In the checkout view the user can enter shipping and billing information, and create the order.

If the user clicks the contact symbol, they will see a page where they can submit a message to the admins of the web application.

The user can also create an account, this is done by clicking on the profile symbol. In this view the user can register a new account, or log in to an already exsisting account.

The user can at any point navigate back to the main page by clicking the Challando logo, or navigate the app with the links in the footer and the navigation bar.

3 Design

As mentioned in the introduction, the two frontend and backend projects are separated and the design of the entire project is as follows:

- Database server that deploy the SQL Derby Databases in which tables and relations are constructed and handled by the application using JPA and Hibernate. The Application connects to the databases using JDBC connection. The application handles 2 databases, one for the main application and the second is to run the DAO unit-tests and the integration tests.
- Data Access Object layer that has the exclusive direct connection to the databases. This layer is included in the model package and consists of the DAO-classes. The DAO-classes are one DAO-class for each entity in the model, where each DAO-class completely handles the persistence of the entity including accessing, adding, removing and changing data from the table(s) in the database that corresponds to the particular entity. The DAO-classes handles also the persistence of the relations between the model-entities. The DAO-test-classes in the test-package provides guarantees that the persistence layer is working as it should.
- The model entities that shapes the skeleton of the application are also included in the model-package. Each entity is persisted by JPA to table

in the database. The attributes are persisted to be columns and the relationships are handled according to the specific need. Some times as extra column in a table and some times as a new table according to the desired functionality of the application.

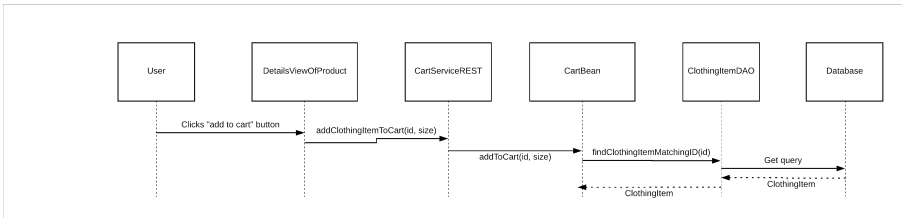
- Business logic layes that uses several DAO-classes to handles some complex tasks such as the login process, the cart tasks and initialisation of the project. Here we have some limitations that we didn't have time ti fix and want to mention here instead: This layer consist of Cartbean, ClothingItemInitBean and CartService. Cartbean, CartItem and ClothingItemInitBean should be separated to business-logic package inside the model but we unfortunately didn't had time for doing that before the deadline. Moreover, Cartbean should be named as SessionHandlingBean because it contains the main functionalities that is provided under the session time such as creating order and not just handling the cart. ClothingItemInitBean also should be called InitBean, which we have in another branch which we worked on but didn't have time to be ready with. This Bean is thought to initialize the entire project with much more records that it has right now. ClothingItemInitBean initialising now categories that makes the name is not suitable. CartService also has business logic functionality for instance in the login endpoints that should be exported to the SessionHandlingBean or another business logic bean in business logic pachage.
- Service layer that expand the functionalities of the DAOs and the business logic layer using JAXRS.
- Security layer that using Java Sequirity API och IdentityStore, the functionality of handling secure login and logout handled by Payara server using the CustomerDAO functionalities to save the data in the database. It could be suitable here to mention that we provide SHA256 encryption to all the Passwords saved in the database from the time they reach the endpoints in the services. They also are validated as SHA256 hash validation. We do the encryption using Apathe DigestUtils.
- The test suit Arquillian is used to provide a complete test environment that mimics the deployment environments to run both the unittests and the integration tests. The integration tests in this project are not cross different micro services or projects but between different DAO-classes that handles entities and relationships. Thus both the unit-tests and the integration tests in this project are focused on the DAO-layer that has the responsibility to persist both the entities and the relationships between the entities. A complete test environment that deploys the project on Arquillian server opening the doors to develop even much more advanced tests that what we had time to do, such as UI-tests and there interaction with the backend.
- Context Dependency Injection and Enterprise Java Bean advantages are

taken all over the backend. All the DAO-classes are initialised as EJB:s in the services and gets the advantages of the EJB transactions where even if the services get many parallel requests, they handles the DAO:s in a thread-safe handling where EJB guarantees no race conditions int the DAO classes. SecurityContext context is injected in 2 places as a CDI bean and ClothingItemInitBean is injected as singletone EJB that guarantees the same instance to be thread-safly invoked in different classes. Magically provides a very nice object-oriented practise throughout the project.

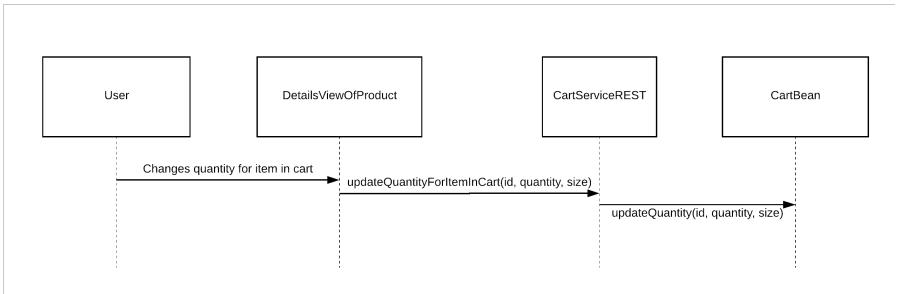
- The frontend is React frontend designed as single page application using react-router. All components that had the same concern are collected in the same packages. Fetch is used to connect to the services in the backend. All the forms are checked and alerts are used to notify the user when invalid input is submitted. Material Design MDBreact framework and ReactBootstrap are used to create the majority of the components that are self responsive. Containers, cards Rows and Cols of ReactBootstrap grid system is mainly used to gain a somehow good responsive design.

4 Application flow

Add clothing item to cart



Change quantity for item in cart



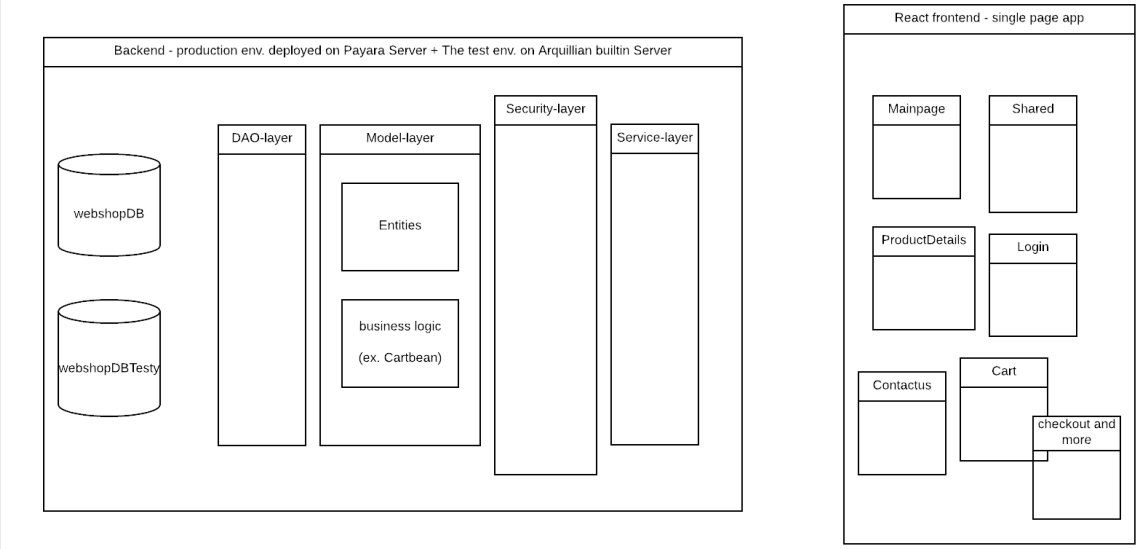


Figure 2: Caption

5 Package diagram

See figure 2.

6 Code Coverage

See figure 3.

7 Model Diagram

See figure 4.

8 Responsibilities

All members have worked on many parts of the project, every member have implemented entities and relationships, and worked with the web services for the resources. Listed below are areas which each member worked on to a greater extent.

Note: The project does not include any of the code from the labs for Madeleine and Philip, instead it's based on the code from Yazan's labs. This is due to the

Figure 3: Image of code coverage report

fact that they were a bit behind on the labs due to sickness, and issues with combining the different labs

8.1 Yazan Ghafir

- login and logout
- signup
- display product in list in the main page
- checkout
- contact us
- validation of all the forms
- Mina sidor
- History
- The layout code of the entire website except the product page and the shopping cart
- alot of work on the DAO, webservices, entities, relationships and security in the backend.

8.2 Madeleine Lexén

- ShoppingCart component

- Product details view
- refactor product to clothingItem in the database
- cartBean
- Cart rest service

8.3 Philip Hellberg

- Persistence
- Troubleshooting, mostly DAOs
- Code Coverage
- Product details view

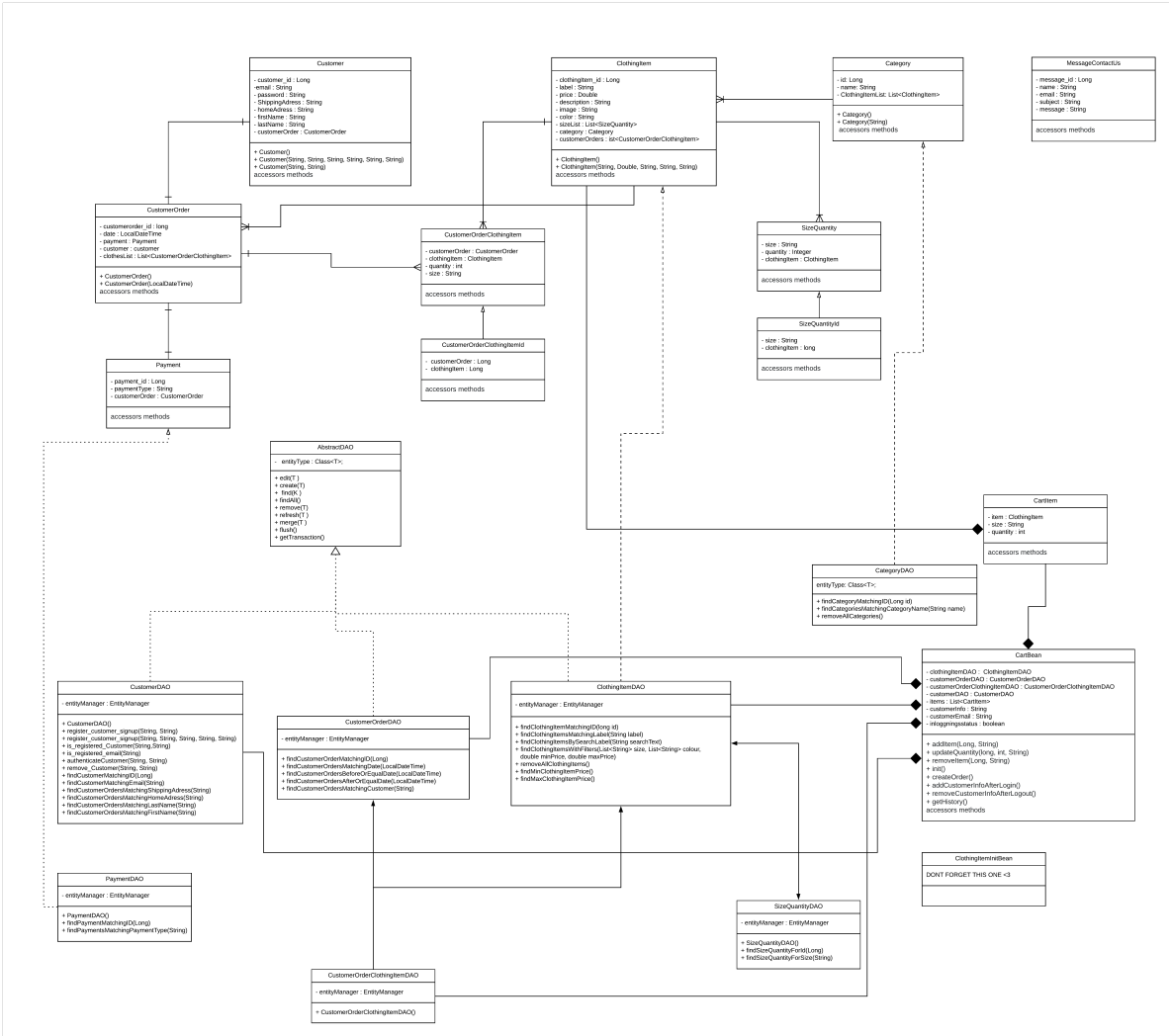


Figure 4: Image of code coverage report